Trường Đại học Khoa học Tự nhiên - ĐHQG-HCM

Khoa Công nghệ Thông tin

BÁO CÁO ĐỒ ÁN 2 - LOGIC

Sinh viên thực hiện: 20120250 - Trần Bảo Anh

Giảng viên: GS.TS Lê Hoài Bắc

Môn học: Cơ sở Trí tuệ nhân tạo

Lớp: 20CNTN

12 - 2022

Mục Lục

1. Tự đánh giá mức độ hoàn thành đồ án

STT	Đặc tả tiêu chí	Tự đánh giá	Điểm
1	Đọc dữ liệu đầu vào và lưu trong cấu trúc dữ liệu phù hợp	Hoàn thành	0.5
2	Cài đặt giải thuật hợp giải trên logic mệnh đề	Hoàn thành	1
3	Các bước suy diễn phát sinh đủ mệnh đề và kết luận đúng	Hoàn thành	2.5
4	Tuân thủ mô tả định dạng của đề bài	Hoàn thành	0.5
5	Báo cáo test case và đánh giá	Hoàn thành	0.5

2. Giải thích mã nguồn

2.1. Cấu trúc dữ liệu

Để lưu trữ một mệnh đề, em sử dụng cấu trúc dữ liệu *list[str]* trong ngôn ngữ Python, với mỗi phần tử trong list là một literal trong mệnh đề. Cụ thể hàm *CLAUSE(raw: str) -> List[str]* sau đây nhận vào một chuỗi thể hiện mệnh đề (ví dụ *'A OR -B OR A'*) và trả về một list các literal tương ứng đã được xử lý để không lặp lại literal và được sắp xếp (*['-B', 'A]*).

```
142    def CLAUSE(raw: str) -> List[str]:
143        return sorted(list(set(raw.replace(' ', '').split('OR'))))
```

2.2. Kiểm tra hai mệnh đề có thể hợp giải hay không

Hàm *RESOLVABLE(a: List[str], b: List[str]) -> bool* nhận vào hai list các literals đại diện cho hai mệnh đề và kiểm tra xem hai mệnh đề có thể hợp giải được hay không. Hai mệnh đề là hợp giải được nếu trong danh sách các literals của mệnh đề b có chứa duy nhất một nghịch đảo của một literal trong mệnh đề a.

```
20    def NEGATIVE(literal: str):
21        if '-' in literal:
22            return literal.replace('-', '')
23        else:
24            return '-' + literal
25
26
27    def RESOLVABLE(a: List[str], b: List[str]) -> bool:
28        ''' Chỉ resolve khi hai mệnh đề có duy nhất
29        một cặp literal đối nhau '''
30        countOppositePair = 0
31        for literal in a:
32            if NEGATIVE(literal) in b:
33                countOppositePair = countOppositePair + 1
34        return countOppositePair == 1
```

2.3. Kiểm tra một mệnh đề có phải là mệnh đề vô giá trị không

Hàm *IS_MEANING_LESS(clause: List[str]) -> bool* nhận vào một danh sách các literals của một mệnh đề và kiểm tra xem mệnh đề đó có phải là một mệnh đề vô giá trị hay không. Một mệnh đề là vô giá trị khi chân trị của nó luôn là *True*, tức chứa ít nhất một cặp literals đối nhau. Ví dụ *'A OR -A'* là một mệnh đề vô giá trị.

```python
def IS_MEANING_LESS(clause: List[str]) -> bool:
    for i in range(len(clause)):
        for j in range(i + 1, len(clause)):
            if NEGATIVE(clause[i]) == clause[j]:
                return True
    return False
```

2.4. Hợp giải hai mệnh đề

Hàm *PL_ RESOLVE(a: List[str], b: List[str]) -> List[List[str]]* nhận vào hai list các literals của hai mệnh đề, tiến hành hợp giải hai mệnh đề nếu được và trả về một danh sách các mệnh đề dưới dạng *List[List[str]]*, do yêu cầu bài toán chỉ hợp giải hai mệnh đề khi chúng có duy nhất một cặp literals đối nhau nên danh sách trả về của hàm này chỉ có tối đa một mệnh đề. Để hợp giải hai mệnh đề, thực hiện các bước như sau:

1. Kiểm tra xem hai mệnh đề có hợp giải được hay không, nếu không thì trả về danh sách rỗng và kết thúc, ngược lại sang bước 2.
2. Hợp hai danh sách literals của hai mệnh đề, loại bỏ literals trùng nhau.
3. Kiểm tra mọi cặp literals trong danh sách ở bước 2, nếu cặp literals đó đối nhau thì đánh dấu chúng để xóa khỏi danh sách.
4. Xóa các literals đã được đánh dấu xóa ở bước 3 khỏi danh sách kết quả.
5. Kiểm tra xem kết quả thu được có phải là một mệnh đề vô giá trị không, nếu có thì trả về danh sách rỗng và kết thúc, ngược lại sang bước 6.
6. Trả về danh sách chứa kết quả.

```python
def PL_RESOLVE(a: List[str], b: List[str]) -> List[List[str]]:
    if RESOLVABLE(a, b):
        data = a + [x for x in b if x not in a]
        for i in range(len(data)):
            for j in range(i + 1, len(data)):
                x = data[i]
                y = data[j]
                if x != 'removed' and y != 'removed' and NEGATIVE(x) == y:
                    data[i] = 'removed'
                    data[j] = 'removed'
        while 'removed' in data:
            data.remove('removed')
        if IS_MEANING_LESS(data):
            return []
        else:
            return [data]
    else:
        return []
```

2.5. Thuật toán kiểm tra suy diễn bằng phương pháp hợp giải

Hàm *PL_RESOLUTION(KB: List[List[str]], alpha: List[str])* nhận vào một danh sách các mệnh đề được cho trong cơ sở tri thức và một mệnh đề alpha. Hàm kiểm tra xem suy diễn từ cơ sở tri thức sang mệnh đề alpha có phải là suy diễn đúng hay không.
Các bước cơ bản của thuật toán như sau:

1. Khởi tạo danh sách *clauses* chứa cơ sở tri thức và mệnh đề alpha (nếu alpha chưa nằm trong cơ sở tri thức).
2. Khởi tạo danh sách *new* bằng rỗng, dùng để chứa các mệnh đề mới được phát sinh trong quá trình hợp giải.
3. Khởi tạo biến số nguyên *start = 0* là vị trí xuất hiện đầu tiên của các mệnh đề mới được thêm vào danh sách *clauses* từ kết quả hợp giải của vòng lặp liền trước đó. Biến này giúp kiểm soát quá trình hợp giải tránh việc hợp giải lại các cặp mệnh đề đã hợp giải trong vòng lặp trước đó.
4. Với mỗi vòng lặp:
   (a) Với mỗi cặp mệnh đề C_i, C_j trong *clauses*, với i chạy từ 0 đến cuối danh sách, j chạy từ *start* đến cuối danh sách, nếu $i \leq j$ thì tiến hành hợp giải hai mệnh đề C_i, C_j và thêm kết quả vào danh sách *new* (loại bỏ mệnh đề trùng nếu có).
   (b) Cập nhật lại biến *start* thành kích thước hiện tại của danh sách *clauses*.
   (c) Nếu danh sách *new* chứa mệnh đề rỗng thì kết luận suy diễn từ KB sang alpha là suy dẫn đúng và kết thúc thuật toán, ngược lại sang bước *(d)*.
   (d) Nếu danh sách *new* là tập con của danh sách *clause* thì kết luận suy diễn từ KB sang alpha là sai và kết thúc thuật toán, ngược lại sang bước *(e)*.
   (e) Thêm danh sách *new* vào cuối danh sách *clauses* (loại bỏ mệnh đề trùng nếu có) và sang vòng lặp tiếp theo.

```python
def PL_RESOLUTION(KB: List[List[str]], alpha: List[str]):
    outputs = []
    clauses = deepcopy(KB)
    clauses = clauses + [x for x in NOT(alpha) if x not in clauses]
    new = []

    loop = 0
    start = 0
    while (True):
        loop = loop + 1
        print('\nloop #' + str(loop), end=': ')

        for i in range(len(clauses)):
            for j in range(start, len(clauses)):
                if i <= j:
                    resolvents = PL_RESOLVE(clauses[i], clauses[j])
                    new = new + [sorted(x) for x in resolvents if sorted(x) not in new]
        newClauses = [sorted(clause) for clause in new if sorted(clause) not in clauses]
        outputs.append(len(newClauses))
        outputs = outputs + [TO_STRING(clause) for clause in newClauses]
        start = len(clauses)
        print(str(len(newClauses)) + ' new clauses.')

        if CONTAINS_EMPTY_CLAUSE(new):
            outputs.append('YES')
            return outputs

        if IS_SUBSET(new, clauses):
            outputs.append('NO')
            return outputs

        clauses = clauses + [sorted(x) for x in new if sorted(x) not in clauses]
```

2.6. Ví dụ

Dữ liệu đầu vào:

- alpha = 'B OR C OR -F'
- KB = ['A OR B OR C', '-E OR -F', 'A OR C OR -D OR -F', 'B OR -E OR -F', '-D']

```
B OR C OR -F
5
A OR B OR C
-E OR -F
A OR C OR -D OR -F
B OR -E OR -F
-D
```

Quá trình hợp giải:

```
========================================================================

Input file: INPUT/input0.txt
Knowledge base: ['A OR B OR C', '-E OR -F', 'A OR C OR -D OR -F', 'B OR -E OR -F', '-D']
Alpha: B OR C OR -F

loop #1
Resolve (A OR B OR C) and (-B) = (A OR C)
Resolve (A OR B OR C) and (-C) = (A OR B)
Resolve (-E OR -F) and (F) = (-E)
Resolve (A OR C OR -D OR -F) and (F) = (A OR C OR -D)
Resolve (A OR C OR -D OR -F) and (-C) = (A OR -D OR -F)
Resolve (B OR -E OR -F) and (F) = (B OR -E)
Resolve (B OR -E OR -F) and (-B) = (-E OR -F)
6 new clauses.

loop #2
Resolve (F) and (A OR -D OR -F) = (A OR -D)
Resolve (-B) and (A OR B) = (A)
Resolve (-B) and (B OR -E) = (-E)
Resolve (-C) and (A OR C) = (A)
Resolve (-C) and (A OR C OR -D) = (A OR -D)
2 new clauses.

loop #3
0 new clauses.

Output: [6, 'A OR C', 'A OR B', '-E', 'A OR C OR -D', 'A OR -D OR -F', 'B OR -E', 2, 'A OR -D', 'A', 0, 'NO']

========================================================================
```

Kết quả:

Đầu vào	**Đầu ra**	**Giải thích**
B OR C OR -F	6	Vòng lặp đầu tiên sinh ra 6 mệnh đề mới
5	A OR C	Hợp giải (A OR B OR C) với (-B) (-B từ phủ định alpha)
A OR B OR C	A OR B	Hợp giải (A OR B OR C) với (-C) (-C từ phủ định alpha)
-E OR -F	-E	Hợp giải (-E OR -F) với F (F từ phủ định alpha)
A OR C OR -D OR -F	A OR C OR -D	Hợp giải (A OR C OR -D OR -F) với F (F từ phủ định alpha)
B OR -E OR -F	A OR -D OR -F	Hợp giải (A OR C OR -D OR -F) với (-C) (-C từ phủ định alpha)
-D	B OR -E	Hợp giải (B OR -E OR -F) với F (F từ phủ định alpha)
	2	Vòng lặp thứ hai sinh ra 2 mệnh đề mới
	A OR -D	Hợp giải (F) với (A OR -D OR -F)
	A	Hợp giải (-B) với (A OR B) hoặc hợp giải (-C) với (A OR C)
	0	Vòng lặp cuối không phát sinh được thêm mệnh đề mới nào
	NO	Kết luận suy diễn sai

3. Nguồn tham khảo

1. Artificial Intelligence: A Modern Approach (Third Edition) - Stuart Russell, Peter Norvig
2. Giáo trình Cơ sở Trí tuệ nhân tạo - Lê Hoài Bắc, Tô Hoài Việt
3. Slide bài giảng được cung cấp trên trang môn học